The RTES Project - BTeV, and Beyond

Michael J. Haney[1], *Member, IEEE*, Shikha Ahuja[2], Ted Bapty[2], Harry Cheung[3], Zbigniew Kalbarczyk[4], Akhilesh Khanna[4], Jim Kowalkowski[3], Derek Messie[5], Daniel Mossé[6], Sandeep Neema[2], Steve Nordstrom[2], Jae Oh[5], Paul Sheldon[7], Shweta Shetty[2], Dmitri Volper[5], Long Wang[4], Di Yao[2]
[1]High Energy Physics, University of Illinois, 1110 W. Green Street, Urbana, IL 61801 USA
[2]Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN 37235 USA
[3]Fermi National Accelerator Laboratory, Batavia, IL 60510 USA
[4]Electrical and Computer Science, University of Illinois, Urbana, IL 61801 USA
[5]Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY 13244 USA
[6]Computer Science, University of Pittsburgh, Pittsburgh, PA 15250 USA
[7]Physics and Astronomy Department, Vanderbilt University, Nashville, TN 37235 USA

Abstract—**The Real Time Embedded Systems (RTES) project was created to study the design and implementation of high-performance, heterogeneous, and fault-adaptive real time embedded systems. The driving application for this research was the proposed BTeV high energy physics experiment, which called for large farms of embedded computational elements (DSPs), as well as a large farm of conventional high-performance processors to implement its Level 1 and Level 2/3 triggers. At the time of BTeV's termination early in 2005, the RTES project was within days of completing a prototype implementation for providing a reliable and fault-adaptive infrastructure to the L2/3 farm; a prototype for the L1 farm had been completed in 2003. This paper documents the conclusion of the RTES focus on BTeV, and provides an evaluation of the applicability of the RTES concepts to other systems.**

Index Terms—**Computer reliability, large-scale systems, real time systems, reliability modeling**

I. Introduction

THE Real Time Embedded Systems (RTES) project [1] was born from a need address the concerns voiced by a project review conducted in 2000, for the proposed BTeV high energy physics experiment [2], which (at the time) called for approximately 2500 embedded processors in its Level 1 trigger, and a comparable number of commodity computers in its Level 2/3 trigger:

> "Given the very complex nature of this system where thousands of events are simultaneously and asynchronously cooking, issues of data integrity, robustness, and monitoring are critically important and have the capacity to cripple a design if not dealt with at the outset… BTeV [needs to] supply the necessary level of "self-awareness" in the trigger system."

The RTES group was formed as a collaborative effort between electrical engineers, computer scientists, and high energy physicists, to research the design and implementation of high-performance, heterogeneous, fault-tolerant and fault-adaptive real-time systems, for which the L1 and L2/3 trigger computational resources of BTeV would serve as a archetype.

Manuscript received June 3, 2005. This work was supported in part by the U.S. National Science Foundation Information Technology Research Program under Grant ACI-0121658.

II. RTES Aspects

The RTES project approached this problem by considering both design-time modeling and run-time capabilities. Figure 1 shows the overall project perspective.

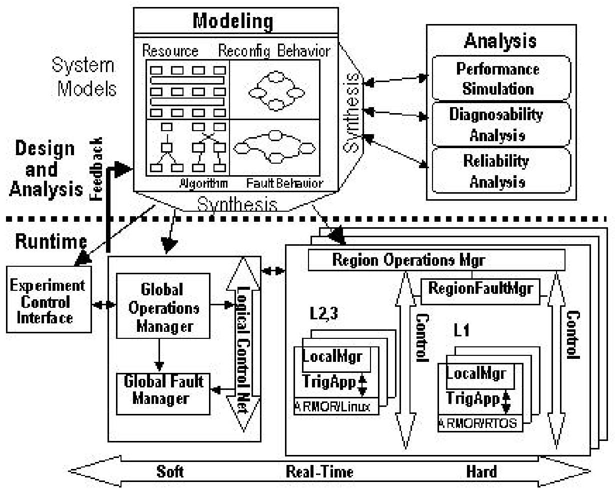

Fig. 1. Design Time and Runtime aspects of RTES. Multiple domain-specific languages are used at the Design and Analysis level, to represent system organization and behavior. Synthesis produces runtime artifacts (codes, scripts). Runtime middleware (VLAs, ARMORs) provides detection and mitigation of faults.

Each of these aspects are described in the following subsections.

A. Modeling

A graphical modeling tool called the Generic Modeling Environment (GME [3]) was used to apply model integrated computing methods to the specification and analysis of the system. GME supports domain-specific languages for representing differing dimensions of the system organization and behavior, as well as the meta-modeling capabilities for defining new domain-specific languages

B. *Adaptive Reconfigurable Mobile Objects for Reliability*

Adaptive Reconfigurable Mobile Objects for Reliability (ARMORs [4]) are multithreaded processes internally structured around objects ("elements") which provide functions or services. Every ARMOR process contains a basic set of elements that provide core functionality, *e.g.*, reliable point-to-point inter-ARMOR messaging, and ARMOR-state checkpointing. A modular, event-driven architecture permits developers to customize an ARMOR process's functionality and fault-tolerance services (detection and recovery) according to the application's needs. The self-checking ARMOR runtime environment includes: one fault-tolerance manager (FTM) to initialize the ARMOR-based system configuration, to maintain registration information on all ARMORs and applications, and to initiate recovery from ARMOR and node failures; one heartbeat ARMOR (HB) to detect failures in the FTM; one daemon ARMOR per node, acting as a gateway for ARMOR-to-ARMOR communication; and any number of execution ARMORs, which launch and monitor application processes.

C. *Very Light Weight Agents*

Very lightweight agents (VLAs [5]) are responsible for providing a lightweight, adaptive layer of fault detection and mitigation. Agents consist of a relatively few lines of code embedded within applications, or acting as independent processes, which monitor hardware and software integrity. VLAs can be proactive or reactive, depending on their scope.

D. *Load Balancing and Network Simulation*

In addition to descriptive system modeling using GME, the RTES project also studied adaptive dynamic load-balancing [6], and thermal management, in an effort to understand the nature of the processing farms, and how best they could be utilized, both to perform their mission-critical processing, as well as to support lower priority (offline) computing on an as-available basis.

III. SUPERCOMPUTING 2003

As a first exercise in demonstrating their methodologies, the RTES group developed a prototype for the BTeV Ll trigger processing farm, using digital signal processors (DSPs) of the type being studied by BTeV at that time. This prototype was demonstrated at the Super Computing 2003 conference [7]. It modeled 3 farmlets of 3 DSPs each, monitored by a PC running Linux; an additional Windows-based PC served as a communications processor. Additional DSPs played the roles of farmlet buffer managers, and the event generator. GME was used to define the execution behavior of the DSPs, as well as to define the communications channels between DSPs and the Windows PC. VLAs were developed for fault detection on the DSPs; ARMORs were not sufficiently "lightweight" for incorporation in the DSP runtime environment. (BTeV later determined that a smaller number of conventional processors, such as the PowerPC, could serve the needs of the L1 trigger as an alternative to DSPs; ARMORs were to have been revisited for incorporation in the next generation L1 farm.) ARMORs were developed for both the Windows and Linux PCs, to provide oversight of the communications and the user control task. Overall control/display was provided by the Experimental Physics and Industrial Control System (EPICS [8]) software. A block diagram of this system is show in Figure 2.

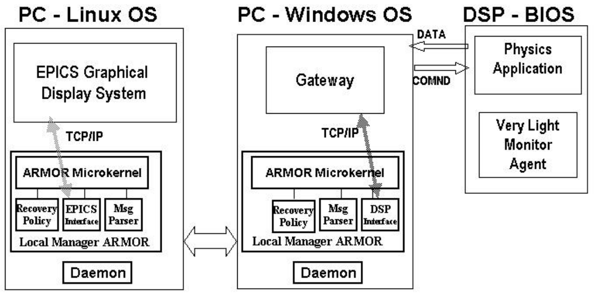

Fig. 2. SC2003 Demonstration System. 16 DSPs were monitored and controlled by a Linux PC, using a Windows PC as a communication channel. VLAs were implemented for the DSPs; ARMORs were developed to support the PCs. VLAs reported to ARMORs.

A formal review of the project software was conducted after the conference. In response to this review, it was recognized that GME would need to serve a large number of domains and submodels: system description, message modeling, fault mitigation behavior, run control behavior, user interface definition, *etc*. For the ARMOR software, it was recognized that custom (application-specific) ARMOR elements needed to be easy to create, and that package organization and version control were vital. This review strongly influenced Demo System 2004.

IV. DEMO SYSTEM 2004

As a next effort, the RTES project undertook to prototype the L2/3 trigger commodity processor farm. Hardware for this farm was accumulated by BTeV, recycled from other computing farms at Fermilab. The farm was heterogeneous (dual-CPU P3's and P4's; at least 4 different speeds), and several exhibited hardware problems. It was an excellent setting for demonstrating reliable software infrastructures.

In accordance with the earlier project review, the use of GME was dramatically expanded in this effort, to provide five different domain-specific graphical languages. The System Integration Modeling Language (SIML) provided a high level specification of the system; artifacts generated from this model (scripts and configuration files) were used to build, deploy, and configure the runtime system. The Data Types Modeling Language (DTML) defined the message data types, and abstracted the details of the underlying communications protocols; artifacts generated from these models (code) provided marshalling and demarshalling middleware. The Run Control Modeling Language (RCML) described the behavior of the underlying trigger application control framework; the artifacts generated from these models (Python scripts) provided global, regional, and local run control state machines. The GUI Configuration Modeling Language (GCML) defined the layout of the user interface; the artifacts generated from this model (Matlab .M files) were used to create the monitoring and control for the demonstration. Matlab was chosen to succeed EPICs for Demo System 2004, with the thought that a commercial-based solution would be easier to develop and maintain. It was also another opportunity to demonstrate the adaptability of GME.

The fifth language addressed a cross-concern, between GME modeling and the ARMOR fault mitigation middleware. The Fault Mitigation Modeling Language allowed finite state machine graphs to characterized custom ARMOR behavior. The artifacts generated from these models were custom ARMOR elements.

Packaging, organization, and versioning concerns were addressed, and resulted in an aggressive use of CVS, and the development of a fully automated build system. Code changes or GME drawing changes, by any of the collaborating team members, could quickly be committed to the shared CVS repository, then checked out and compiled by another. Since GME defined metamodels for the graphical languages, it was commonplace for a metamodel to "compile" into an interpreter, which in turn processed a domain model, to produce C++ source code, which was then compiled into the run tree.

A block diagram of the system is shown in Figure 3.

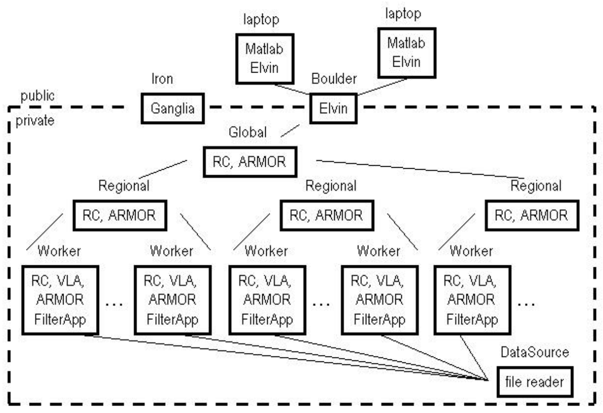

Fig. 3. Demo System 2004. Worker nodes running the L2/3 filter application, with Run Control (RC), VLAs and ARMOR middleware, are overseen by regional manager nodes (also ARMORed), which are overseen by an ARMORed global node. The user interface is GME-define Matlab code. Elvin publish-subscribe messaging is used for non-ARMOR communications.

Several different test configurations were developed using the SIML system language, employing variously 3, 12, and 54 worker nodes (performing L2/3 processing), with additional regional and global control nodes. As each node was a dual-CPU machine, these configurations allowed the testing of ARMORs, VLAs, and GME-derived communications and control to be applied to over 120 processors. The "16 node" (12 workers) and "65 node" (54 workers) systems were demonstrated at the Real Time and Embedded Technology and Applications Symposium, March 2005 [9].

Developing and supporting both "16 node" and "65 node" configurations had several beneficial effects. Effective software engineering was vital to minimize the number of instances where the "same change" needed to be applied. And rapid remapping between configurations exposed scale-dependent behaviors and system bugs that might not otherwise have been detected only by the testing of one configuration.

V. Beyond BTeV

Beyond the BTeV project, the RTES project is examining several alternative experiments and projects to which the solutions developed for and lesson learned from the SuperComputing 2003 and Demo System 2004 prototypes may be applied.

A. CMS

Model integrated computing and domain-specific modeling languages, with automated code generation, are applicable to any large scale system, to mitigate complexity associated with design management and component integration.

A study is in progress in the use of GME-based models for system description, finite state machine representation, and message definition, each with its own automated artifact generation, for potential application to the XDAQ subsystem of the CMS project at CERN. Each domain-specific graphical language has an associated interpreter, which produces scripts (*e.g.* Python), code fragments (*e.g.* C++), configuration files, *etc*., as appropriate for the problem domain.

The HLT (high level trigger) framework is also being explored, as a candidate for adding fault tolerance modules. Several VLA design principles are being investigated for providing adaptive capabilities under dynamic error conditions, including game theoretical solutions assuming rational agents [10].

As CMS is a well developed project with critical deadlines, the RTES study is seeking to demonstrate solutions to outstanding needs, without requiring changes or resources from CMS. Unlike BTeV, where model integrated computing had an opportunity to define the methodologies employed in system design, RTES must now demonstrate agility and economy in addressing existing issues in CMS. However, this demonstration should prove general applicability of these RTES concepts.

B. LQCD

Many computational systems with fault sensitivities can benefit from automated detection and fast fault recovery.

The ARMOR middleware is being ported to the Lattice Gauge Theory Computational Facility at Fermilab, in support of Lattice Quantum Chromo Dynamics (LQCD) calculation. This processing is highly sensitive to faults, as the failure of a single process in an active cluster can compromise the processing of the entire cluster. Check-pointing and real time process recovery/migration are being studied, to mitigate the system-wide effects of single point failures. The LQCD codes are real applications, unlike the L2 model used in Demo System 2004. Also, varying numbers of protected nodes will create an interesting testbed for ARMOR configuration and communication scaling.

The LQCD applications further differ from the L1 and L2/3 farm applications of BTeV, in that they are batch processing (rather than operator overseen), and the scope of processes to be supervised is considerably more dynamic than the quasi-static L1 and L2/3 trigger applications. Fully self-sufficient solutions, with clean consumer (API) interfaces, are required.

C. *Grid-based applications*

Load balancing and networks studies of the BTeV L2/3 farm will be generalized, to consider inter-farm scheduling and communications, in support of Grid-based processing. The farms provide not only computational power for the reconstruction tasks, but also off-line and analysis jobs. This expansion will ensure that the scheduling of the farm resources will be done in a more resource-efficient manner, taking into account the resource needs of the jobs (*e.g.* number of processors or processor-hours required), deadlines (*e.g.* "need result by Monday noon"), as well as load, temperature, and energy constraints.

D. *Other*

Another testbed being examined for ARMORs and VLAs is the Dark Energy Survey (DES), and their adoption of the MONSOON data acquisition system with a digital camera to be used on the Blanco 4m telescope at the Cerro Tololo Interamerican Observatory. The site is remote (a hilltop in Chile), so inference and adaptation must replace observation and intervention.

This is not a critical, real time hard system, as it may be possible to repeat an image acquisition if a fault is quickly detected and remitted. Also, the system is computationally modest, requiring only a few processors. But the application software for this project is well developed and documented, and will clearly demonstrate the "cost" of incorporating ARMOR and VLA methodologies.

VI. CONCLUSION

The RTES project was created to address the fault adaptive needs of the BTeV high energy physics project. However, the design-time modeling and runtime middleware developed by this project are applicable to many large, high performance, heterogeneous, real-time embedded application environments. Several differing environments are currently being explored.

REFERENCES

[1] Information on the RTES project is available from www-btev.fnal.gov/public/hep/detector/rtes/index.shtml

[2] Information on the BTeV Experiment is available from www-btev.fnal.gov/public/GeneralInformation.shtml

[3] A. Ledeczi, M. Maroti, A. Bakay, G. Karsai, J. Garrett, C. Thomason IV, G. Nordstrom, J. Sprinkle, P. Volgyesi, "The Generic Modeling Environment", Workshop on Intelligent Signal Processing, Budapest, Hungary, May 17, 2001.

[4] Z. Kalbarczyk, R. K. Iyer, and L. Wang, "Application Fault Tolerance with Armor Middleware," *IEEE Internet Computing*, Special Issue on Recovery-Oriented Computing, March/April 2005, pp 28-37.

[5] J. C. Oh, M. S. Tamhankar, D. Mosse', "Design of Very Lightweight Agents for Reactive Embedded Systems", *Proceedings of the 10th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, 7-10 April 2003, pp 149-158.

[6] J. Oh, R. Chopade, S. Vagir, R. Garg, "RK+MOSIX: A Real-Time Kernel with Task Migration Support", Brazilian Workshop of Real-Time, Fortaleza, Brazil, 13 May 2005.

[7] D. Messie, M. Jung, J. C. Oh, S. Shetty, S. Nordstrom, M. Haney, "Prototype of Fault Adaptive Embedded Software for Large-Scale Real-Time Systems", *Proceedings of the 12th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, 4-7 April 2005, pp 498-505.

[8] L. Dalesio, "EPICS: Recent Applications and Future Directions", *Proceedings of the 2001 Particle Accelerator Conference*, 18-22 June, 2001, pp 276-278.

[9] S. Ahuja, et al., "RTES Demo System 2004", *ACM SIGBED Review*, Special Issue on High Performance, Fault Adaptive, Large Scale Embedded Real-Time Systems, July 2005, Volume 2, Number 3.

[10] D. Messie, and J. C. Oh, "Polymorphic Self-* Agents for Stigmergic Fault Mitigation in Large-Scale Real-Time Embedded Systems", Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), Utrecht, The Netherlands, July, 2005.